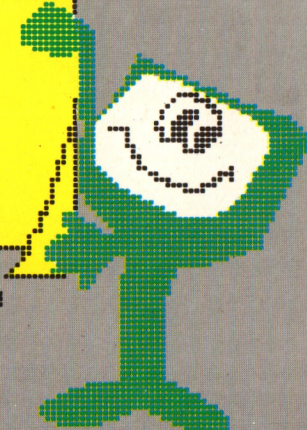
20 VIDEOLEZIONI DI BASIC
PER IMPARARE CON L'MSX

VIDEOBASIC MSX
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea -
Via Indipendenza 88-90 - Como
Redazione software:
Michele Casartelli
Francesco Franceschini
Progetto grafico:
Studio Nuovidea - via Longhi, 16 - Milano
Impaginazione:
Moreno Confalone
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70 (autorizzazione della Direzione Provinciale delle PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson S.r.l. - Via Rosellini, 12 20124 Milano, mediante emissione di assegno bancario o cartolina vaglia oppure utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere richiesti direttamente all'editore inviando L. 10.000 cdu. mediante assegno bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

SOMMARIO

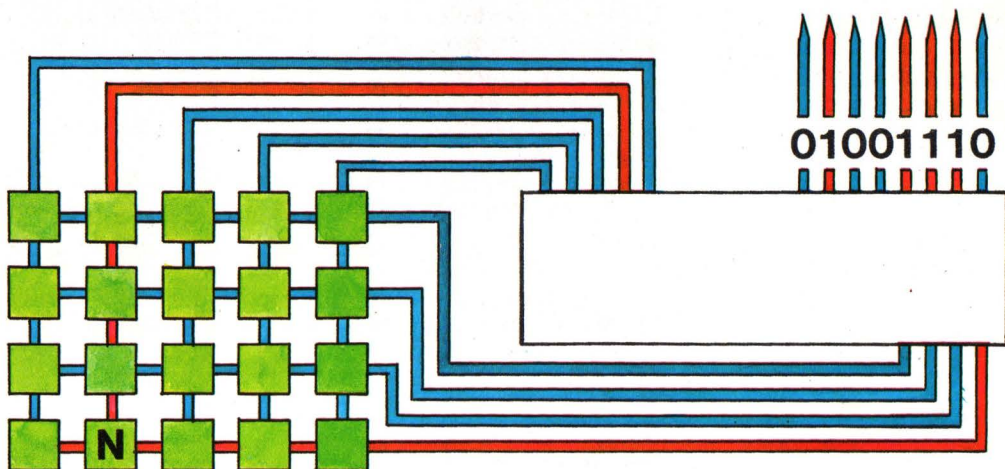
Introduzione

In questa lezione approfondiremo la conoscenza della tastiera, il principale dispositivo di ingresso dei dati.
Non tutte le tastiere sono però uguali, i meccanismi variano, infatti, da un tipo all'altro così come le caratteristiche non che il principio di funzionamento.
Indissolubilmente legati alla tastiera sono il codice ASCII e il set dei caratteri.
Impareremo poi, a conoscere e usare, le ASC - CHR$ - INKEY$ - KEY ON - KEY OFF e FOR-TO - STEP-NEXT, che ti permetteranno di eseguire quante volte vuoi un gruppo di istruzioni.
Per finire, una tecnica indispensabile a ogni programmatore: i cicli automatici.

Schema e funzionamento dei tipi di tastiera

La tastiera costituisce sicuramente il principale dispositivo di ingresso delle informazioni di cui è fornito il tuo computer. È fondamentalmente attraverso essa che ti è infatti possibile comunicare all'unità centrale tutti i comandi, le istruzioni ed i dati che intendi eseguire o memorizzare.
Un computer senza tastiera è come una automobile senza volante: lo potresti mettere in moto ed arrestare, ma non controllare ed utilizzare.
È pertanto importante che tu capisca, al di là del semplice e consueto utilizzo di tutti i giorni, la struttura ed il principio di funzionamento della tastiera di un elaboratore.
Prima di affrontare il discorso è però necessario precisare e chiarire con esattezza cosa si intende con il termine «tastiera». Tale parola specifica infatti solo ed esclusivamente il dispositivo utilizzato per l'ingresso dei dati.
Chiamare «tastiera» un intero calcolatore (come fanno alcune persone non molto informate) è assolutamente sbagliato e scorretto!
Come hai potuto verificare, la tastiera del tuo MSX è sostanzialmente identica, nell'aspetto e nel funzionamento, a quella di una comune macchina da scrivere: basta schiacciare il tasto corrispondente al carattere prescelto ed il gioco è fatto.
Eventualmente, mediante la combinazione di due o tre tasti premuti in contemporanea, si possono comporre ulteriori lettere, simboli o comandi che di solito non sono disponibili sulle normali macchine da scrivere.
Una particolarità che forse ti è sfuggita è la disposizione delle lettere: esse sono infatti ordinate secondo lo standard statunitense, chiamato QWERTY.
Questo nome, assegnato alle tastiere di tipo americano, nasce proprio dalla collocazione dei primi sei tasti alfabetici della seconda fila di tasti.
Nelle tastiere cosiddette europee, invece, la Z è in seconda posizione al posto della W; da qui il nome QZERTY.
Ulteriori differenze sono la posizione della M e la disposizione di quasi tutti i simboli e segni di punteggiatura.
Nulla cambia comunque agli effetti pratici: entrambe le tastiere (americana ed europea) si comportano in modo assolutamente identico ed affidabile ai fini del funzionamento.
Cerchiamo ora di capire come effettivamente

funziona una tastiera, cioè cosa succede quando premi un tasto del tuo MSX.
Tutti i tasti presenti sulla tastiera sono connessi elettricamente (cioè tramite fili conduttori di elettricità) ad un particolare circuito integrato, che ne rileva l'effettivo azionamento e produce per ciascuno dei tasti premuti un unico e distinto codice numerico binario a 8 bit.
L'unità centrale, quando riceve una simile combinazione, è così immediatamente in grado (normalmente per mezzo di un programma memorizzato su ROM o di un ulteriore circuito elettronico) di distinguere ed individuare il particolare carattere premuto, in modo da eseguire le operazioni richieste.
Così, per esempio, quando premi la lettera A, all'uscita del circuito di codificazione (questo è il termine tecnico usato per indicare tale componente) compare il codice binario 01000001, corrispondente a 65 in decimale. A tale codice (e solo ad esso!) corrisponde per la CPU il carattere A: non vi è quindi alcuna possibilità che nella macchina insorgano confusioni ed errori.

Un circuito integrato individua e riconosce il tasto premuto ed emette il codice binario corrispondente.

HARDWARE

Il codice ASCII

I codici numerici da assegnare a ciascuno dei caratteri più utilizzati non vengono scelti arbitrariamente dalla casa costruttrice, ma sono il frutto di una cooperazione avvenuta tra utenti di apparecchiature ed industrie operanti nel ramo della elaborazione dei dati. In origine tali codici erano stati infatti realizzati con lo scopo di semplificare e standardizzare le comunicazioni tra i diversi calcolatori, eliminando così tutti i problemi connessi a differenti rappresentazioni dei dati o delle informazioni.
La diffusione sempre più ampia dei personal computer ha fatto sì che

Decimale	ASCII	Decimale	ASCII	Decimale	ASCII
0	NUL	43	+	86	V
1	SOH	44	,	87	W
2	STX	45	–	88	X
3	ETX	46	.	89	Y
4	EOT	47	/	90	Z
5	ENQ	48	0	91	[
6	ACK	49	1	92	\
7	BEL	50	2	93	]
8	BS	51	3	94	^
9	HT	52	4	95	_
10	LF	53	5	96	`
11	VT	54	6	97	a
12	FF	55	7	98	b
13	CR	56	8	99	c
14	SO	57	9	100	d
15	SI	58	:	101	e
16	DLE	59	;	102	f
17	DC1	60	<	103	g
18	DC2	61	=	104	h
19	DC3	62	>	105	i
20	DC4	63	?	106	j
21	NAK	64	@	107	k
22	SYN	65	A	108	l
23	ETB	66	B	109	m
24	CAN	67	C	110	n
25	EM	68	D	111	o
26	SUB	69	E	112	p
27	ESC	70	F	113	q
28	FS	71	G	114	r
29	GS	72	H	115	s
30	RS	73	I	116	t
31	US	74	J	117	u
32	spazio	75	K	118	v
33	!	76	L	119	w
34	"	77	M	120	x
35	#	78	N	121	y
36	$	79	O	122	z
37	%	80	P	123	{
38	&	81	Q	124	|
39	'	82	R	125	}
40	(	83	S	126	~
41	)	84	T	127	DEL
42	*	85	U		

tale codificazione, chiamata ASCII (abbreviazione di American Standard Codes for Information Interchange, cioè codici standard americani per l'interscambio delle informazioni), sia diventata di fatto uno "standard" presente nella totalità dei computer. I caratteri alfabetici e di punteggiatura presenti sul tuo MSX, sono quindi composti con le stesse combinazioni di bit codificate negli altri calcolatori quando su di essi vengono premuti i tasti corrispondenti.

Tasti e tastiere

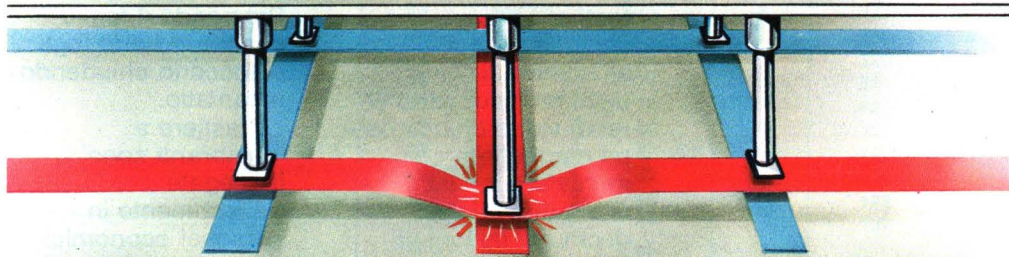
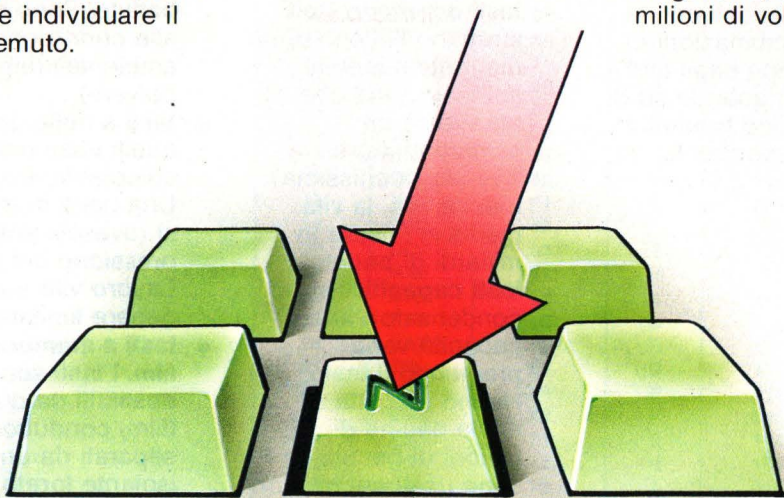
La tastiera è sottoposta a continue sollecitazioni meccaniche. La sua vita (o durata) dipende in gran parte dalla qualità dei tasti: può andare da qualche decina di migliaia di battute (nelle tastiere veramente scadenti) a molte decine di milioni.
Vediamo succintamente i vari tipi di tasti, a partire dai migliori:

- **tasti ad effetto Hall**: sfruttano l'effetto di un magnetino mobile sulla corrente che attraversa un semiconduttore. Avendo pochissima meccanica, la vita media si misura in miliardi di battute;
- **tasti capacitivi**: sono condensatori la cui capacità varia premendo il tasto. Hanno una vita di molte decine di milioni di battute e sono usati nei migliori personal computer;
- **tasti a reed**: un contatto posto all'interno di un'ampolla di vetro (il reed) viene chiuso da un magnetino montato sul nastro. La vita è di qualche decina di milioni di battute. A causa della concorrenza dei più robusti tasti capacitivi, il loro uso si è molto ridotto negli ultimi anni;
- **tasti meccanici standard**: sono impiegati da quasi tutti i personal computer, compreso il tuo MSX. La loro vita dipende dalla qualità costruttiva. Nel caso migliore è di alcune decine di milioni di battute. Sono sensibili alle condizioni ambientali (umidità, polvere);
- **tasti a bolla**: sono quelli usati nelle calcolatrici tascabili. Una bolla di metallo si rovescia sotto la pressione del tasto. La loro vita è in genere limitata;
- **tasti a membrana o a film**. I tasti sono costituiti da due fogli (film) conduttori, tesi e separati da un foglio isolante forato nella forma adatta. Premendo il foglio superiore i due fogli si toccano chiudendo il contatto.
Le tastiere a membrana sono impiegate generalmente in personal economici od in applicazioni

industriali, in quanto garantiscono un ottimo isolamento dai fattori esterni che possono pregiudicarne il funzionamento. La durata dipende dalla costruzione.

A parte le differenze costruttive, comunque, possiamo ricondurre a una sola sequenza le operazioni che esegue il tuo computer, o meglio la CPU per analizzare la tastiera e individuare il tasto premuto.

Vediamo in modo estremamente semplificato cosa accade:

1. periodicamente, e ad intervalli stabiliti, la CPU interrompe quello che sta facendo per rivolgere la sua attenzione alla tastiera;
2. esegue la cosiddetta routine di interrupt, memorizzata nella ROM;
3. controlla (in gergo, scanning) lo stato della tastiera, cioè se un tasto è stato premuto;
4. se questo è avvenuto, la CPU ricava la posizione del tasto stesso;
5. una volta che ha verificato il punto 4 ricontrolla una frazione dopo (ricorda che la velocità delle CPU è dell'ordine dei megahertz, cioè milioni di volte al

secondo), in modo da escludere eventuali interferenze;

6. a questo punto la CPU ritorna al suo lavoro, per riprendere poi la sequenza illustrata.

Per concludere. La tastiera è il principale mezzo di comunicazione tra te e il computer: trattala con cura.
Essendo infatti meccanica, è soggetta ad usura e la durata della sua vita dipende dall'uso che ne fai.
Evita perciò colpi bruschi, pressioni esagerate dei tasti e soprattutto ... quando un programma "non gira" evita di sfogare sulla tastiera la tua rabbia.

Il set dei caratteri

Come ben sai (scusa la pedanteria, ma occorre) il tuo MSX come qualsiasi altro computer, sa usare e memorizzare solo numeri, per di più solo numeri binari.
Come fa allora a capire e visualizzare sul video il punto di domanda (?), la scritta "ciao", l'asterisco (*)?
Semplice. Il tuo MSX trasforma in numeri tutti i caratteri alfabetici, numerici e speciali (il set di caratteri della macchina) che è in grado di gestire, inviare e ricevere.
Per questa codifica utilizza un codice assai simile a quello usato da tutti gli altri personal computer: il citato codice ASCII.
È per questo che il tuo computer sa "manipolare" anche stringhe: le interpreta come una successione di numeri codificati corrispondenti ad una precisa tabella di caratteri presente nella memoria.
Quello che è importante è che a ogni codice corrisponde uno e un solo carattere, in modo che il computer non abbia mai alcuna ambiguità di interpretazione.
Il set dei caratteri del tuo MSX utilizza un codice a 8 bit; sono perciò possibili 256 (cioè 2^8) combinazioni, alle quali corrispondono altrettanti caratteri. Molti di questi caratteri sono immediatamente e facilmente riconoscibili anche alla prima occhiata, essendo di frequente utilizzo per chiunque: lettere alfabetiche, caratteri di punteggiatura, cifre numeriche.
Altri invece, prima di assumere un qualsiasi significato, richiedono un attimo di riflessione; altri, i caratteri grafici,

opportunamente impiegati, consentono di comporre facilmente schermate in grado di rappresentare più efficacemente che con le parole particolari situazioni da visualizzare.

STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$
	32	;	59	W	87	r	114
!	33	<	60	X	88	s	115
"	34	=	61	Y	89	t	116
#	35	>	62	Z	90	u	117
$	36	?	63	[	91	v	118
%	37	@	64	\	92	w	119
&	38	A	65	]	93	x	120
'	39	B	66	^	94	y	121
(	40	C	67	_	95	z	122
)	41	D	68	`	96	{	123
*	42	E	69	a	97	|	124
+	43	F	70	b	98	}	125
,	44	G	71	c	99	~	126
-	45	H	72	d	100		127
.	46	I	73	e	101	Ç	128
/	47	J	74	f	102	ü	129
0	48	K	75	g	103	é	130
1	49	L	76	h	104	â	131
2	50	M	77	i	105	ä	132
3	51	N	78	j	106	à	133
4	52	O	79	k	107	å	134
5	53	P	80	l	108	ç	135
6	54	Q	81	m	109	ê	136
7	55	R	82	n	110	ë	137
8	56	S	83	o	111	è	138
9	57	T	84	p	112	ï	139
:	58	U	85	q	113		
		V	86				

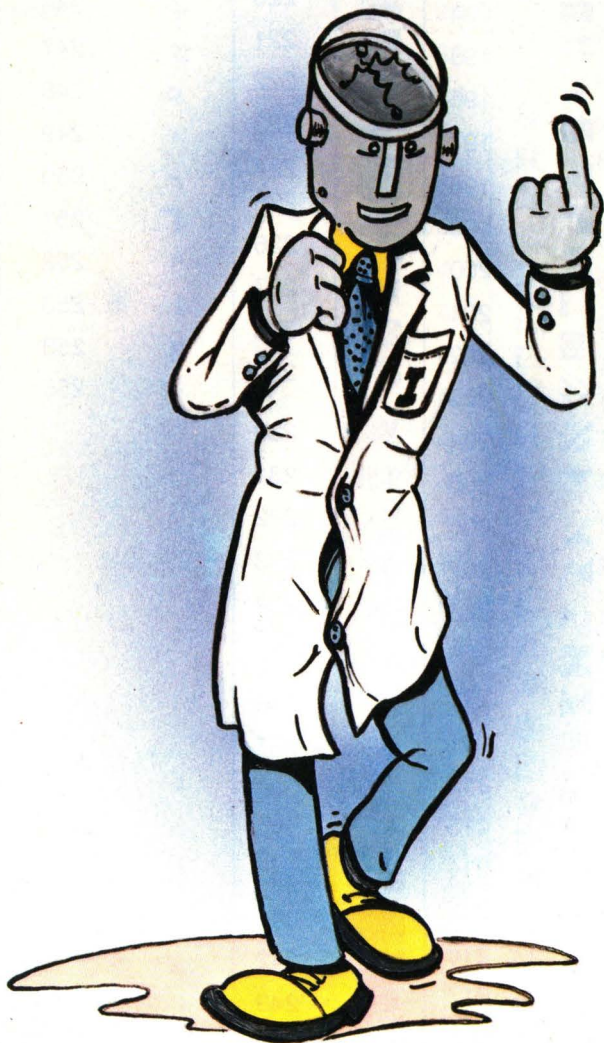
STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$
î	140	ª	166		192		218	⌠	244
ì	141	º	167		193		219	⌡	245
Ä	142	¿	168		194		220	÷	246
Å	143	⌐	169		195		221	≈	247
É	144	¬	170		196		222	°	248
æ	145	½	171		197		223	∙	249
Æ	146	¼	172		198	α	224	·	250
ô	147	¡	173		199	β	225	√	251
ö	148	«	174		200	Γ	226	ⁿ	252
ò	149	»	175		201	π	227	²	253
û	150	Ã	176		202	Σ	228	■	253
ù	151	ã	177		203	σ	229		255
ÿ	152	Ĩ	178		204	µ	230		
Ö	153	ĩ	179	▼	205	τ	231		
Ü	154	Õ	180	▲	206	Φ	232		
¢	155	õ	181	►	207	Θ	233		
£	156	Ũ	182	◄	208	Ω	234		
¥	157	ũ	183		209	δ	235		
Pt	158	IJ	184		210	∞	236		
ƒ	159	ij	185		211	φ	237		
á	160	¾	186		212	∈	238		
í	161	~	187		213	∩	239		
ó	162	◊	188		214	≡	240		
ú	163	‹	189		215	±	241		
ñ	164	¶	190	▲	216	≥	242		
Ñ	165	§	191	‡	217	≤	243		

ASC

A volte può essere utile conoscere il codice ASCII di un dato carattere o, viceversa, essere in grado di produrre o stampare un certo carattere dato il suo codice. Il BASIC mette a tua disposizione due istruzioni, attraverso cui puoi convertire i caratteri in codici ed i codici in caratteri: ASC e CHR$. Entrambe queste istruzioni sono delle funzioni; è quindi necessario che a ciascuno di esse tu fornisca un argomento sul quale operare.

ASC produce il valore del codice ASCII, cioè un numero corrispondente al primo carattere di una stringa. L'argomento deve pertanto essere una stringa sotto forma di costante (racchiusa naturalmente tra virgolette) o di variabile. ASC fornisce come risultato un numero compreso tra 0 e 255: i caratteri disponibili sul tuo computer sono infatti, come detto, complessivamente 256. Se l'argomento è una stringa nulla, ASC provoca il messaggio di errore ? Illegal function call.
Vediamo qualche esempio:

```
PRINT ASC ("A")
```

Ottieni 65, il codice numerico ASCII del carattere "A"

Sintassi della funzione

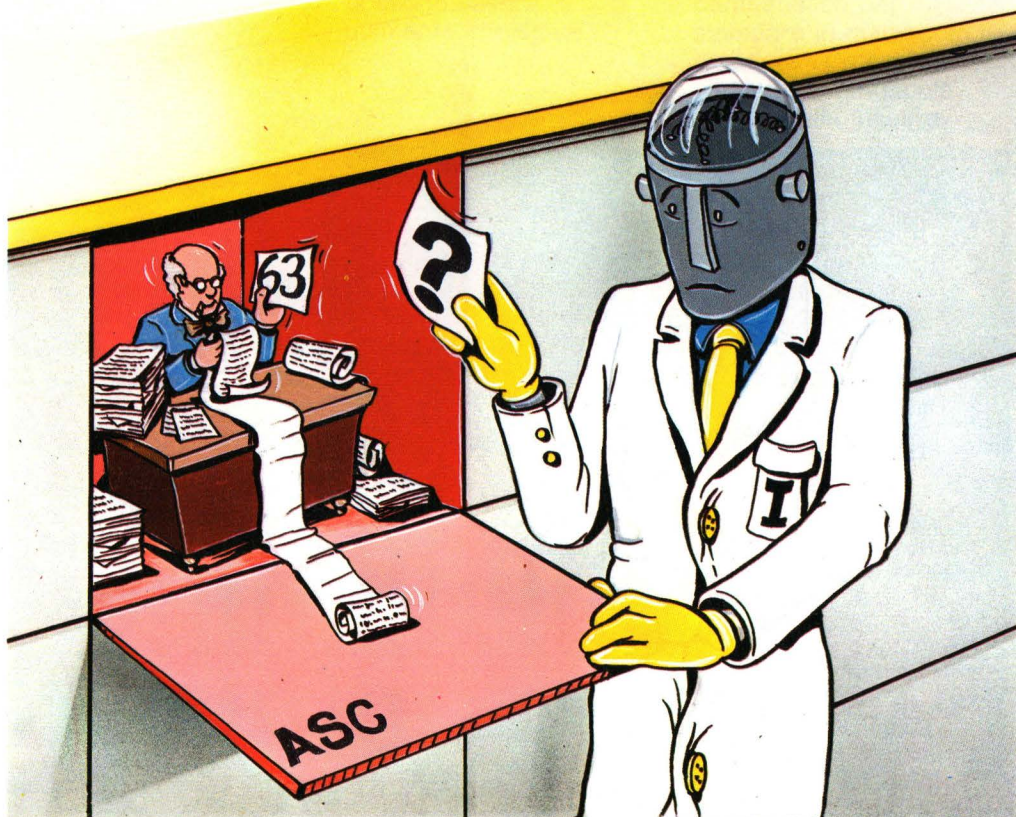
ASC (stringa)

L'argomento di ASC è un carattere. Il risultato è il suo codice numerico.

```
PRINT ASC ("ABCD")
```

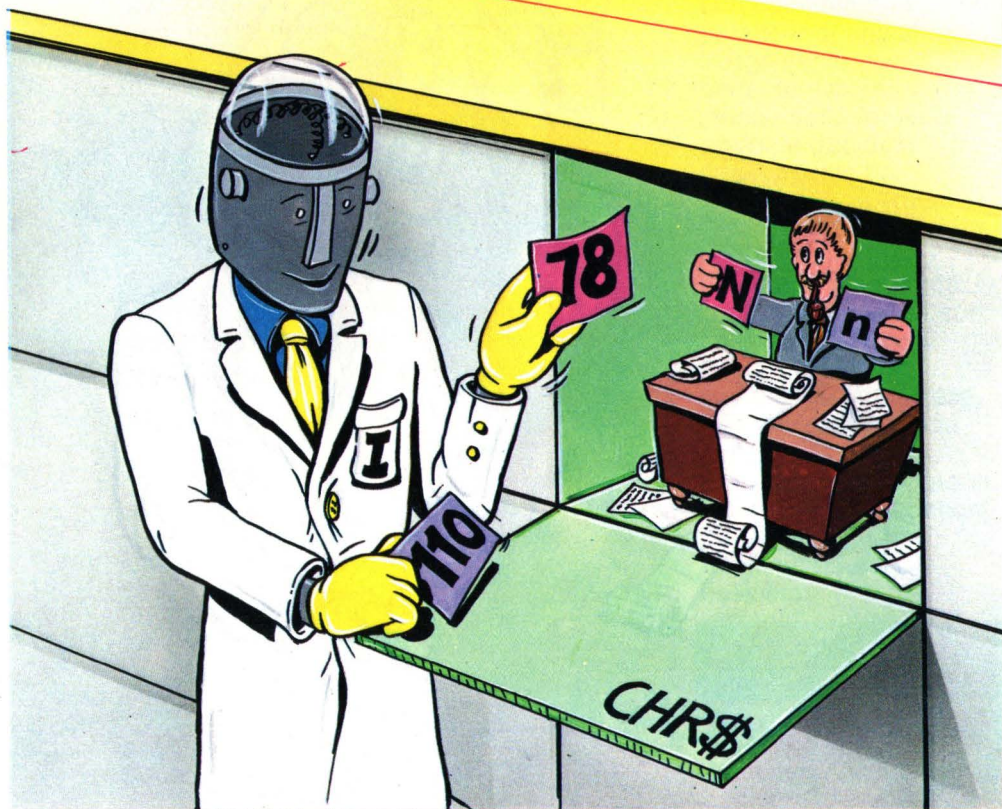
Ottieni 65: il primo carattere dell'argomento è "A"

```
50 C$ = "TAVOLO"
60 PRINT ASC (C$)
```

Ottieni 84, codice numerico del carattere "T"

CHR$

CHR$ è, in un certo senso, la funzione opposta di ASC: essa ti restituisce il carattere corrispondente al numero che avrai utilizzato come argomento.
Tale numero può essere specificato sia per mezzo di una variabile che di un'espressione.
L'argomento deve essere compreso tra 0 e 255; in caso contrario l'elaboratore visualizzerà il messaggio di errore ? Illegal function call.
Quando il valore dell'argomento è un numero decimale, CHR$ lo tronca al valore intero immediatamente inferiore:

```
15 PRINT CHR$(151/2)
```

151 / 2 = 75.5: il numero intero immediatamente più piccolo è 75. A tale argomento corrisponde il

carattere K.
Il seguente programma stampa l'intero set dei caratteri disponibili sul tuo MSX:

```
10 FOR A=32 TO 255: PRINT A, CHR$ (A): NEXT A
```

Se desideri controllare i codici ed i rispettivi caratteri, utilizza il tasto stop per interrompere e riprendere l'esecuzione del programma.
Le funzioni ASC e CHR$ possono essere utilmente impiegate quando, per esempio, si vuole trasformare una lettera maiuscola nel corrispondente carattere minuscolo (o viceversa). Analogamente a quanto stabilito dal codice ASCII, le maiuscole sono codificate con numeri minori delle minuscole, e la differenza di codice tra la stessa lettera minuscola-maiuscola è uguale a 32.
Il programma che segue sfrutta proprio questa caratteristica per stampare in maiuscolo il carattere corrispondente alla lettera minuscola battuta sulla tastiera

```
 5 REM attenzione al CAPS LOCK : non deve
   essere attivo !
10 CLS
20 INPUT A$ : IFA$="*" THEN END
30 IF ASC (A$) < 97 THEN GOTO 20
40 IF ASC (A$) > 122 THEN GOTO 20
50 PRINT A$, CHR$ (ASC (A$) — 32) : REM stampa
   i carattere minuscolo ed il corrispondente
   maiuscolo
60 GOTO 20
```

Esempi

```
PRINT CHR$(65)
```

Il codice 65 corrisponde al carattere "A".

```
PRINT CHR$ (18 * 5)
```

L'argomento della funzione CHR$ può essere una espressione il cui valore è compreso tra 0 e 255.

L'argomento di CHR$ è un codice numerico, il suo risultato è il carattere corrispondente.

```
S = (100 - 1)
PRINT CHR$ (S/3)
```

Se vai a vedere la tabella ASCII pubblicata qualche pagina indietro vedrai che al codice 33 corrisponde il punto esclamativo (!).

```
PRINT CHR$ (65.91)
```

Quando l'argomento di CHR$ è un numero decimale, viene troncato al valore intero immediatamente più piccolo, in questo caso 65.
Si ottiene perciò la stampa del carattere A.

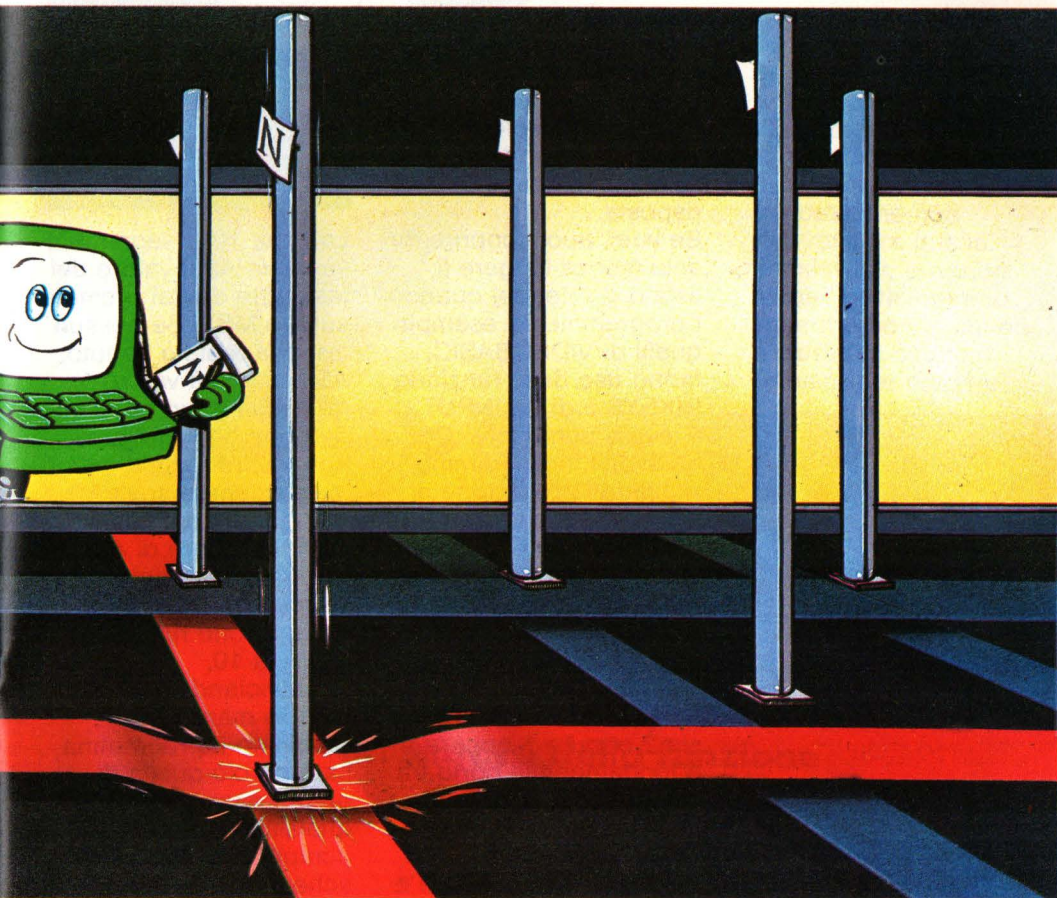
Sintassi della funzione

CHR$ (numero)
dove NUMERO può essere un numero, una variabile od un'espressione numerica

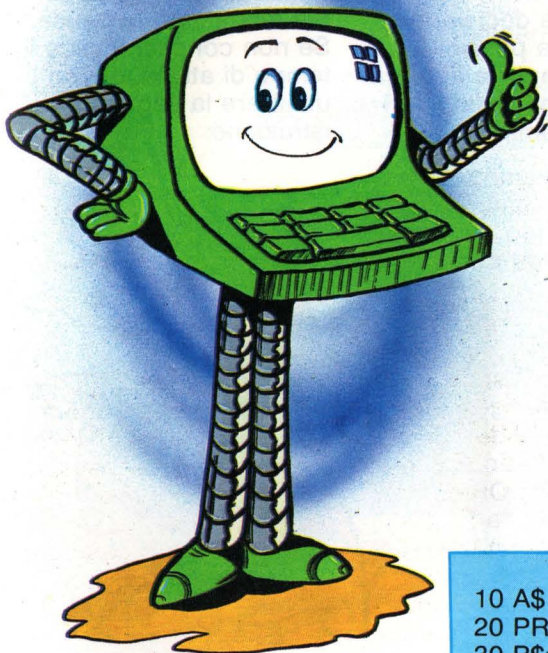
INKEY$

LINGUAGGIO

L'istruzione INKEY$ consente l'ingresso dalla tastiera del tuo MSX di un singolo carattere, senza però dover battere anche il tasto RETURN. Il carattere corrispondente al tasto premuto non viene inoltre visualizzato sullo schermo.
Perché non usare il già familiare e ormai conosciuto INPUT, allora?
Non è forse vero che anche con INPUT si può introdurre un solo carattere? La differenza è sottile, ma importante.
Come visto parlando di INPUT, se sbagli il dato da introdurre, le conseguenze possono essere catastrofiche: risultati sbagliati o, peggio, programma bloccato.
Con INKEY$, nulla di tutto questo: è quasi

Quando incontra INKEY$ il tuo MSX annota il carattere che stai digitando in quel momento.

fatto apposta per permetterti di sbagliare (input controllato) senza che succeda nulla. Anzi, il tuo MSX attende tranquillo che tu introduca esattamente la risposta che si aspetta da te.
Il termine "attende" merita una considerazione a parte. Con INPUT il programma viene interrotto per aspettare i dati in ingresso, INKEY$ invece viene eseguito come una normale funzione BASIC, alla velocità di cui è capace l'interprete e quindi ben superiore a quella di qualsiasi uomo. Occorre, perciò, inserire INKEY$ in un ciclo di attesa, che renda "umano" il tempo di risposta.
Se vuoi, puoi scoprire da solo (senza leggere il listato si intende) quando i programmi, ad esempio quelli di VIDEOBASIC, fanno uso della funzione INKEY$ o di un INPUT. Prova a pensarci ... Ogni qualvolta il programma richiede la pressione di un tasto seguito da RETURN vuol dire che sotto c'è INPUT. Altrimenti ... L'argomento di INKEY$ può essere una variabile stringa. Quindi, se vuoi che il carattere riconosciuto sia 2 oppure * oppure !, è fondamentale porlo tra virgolette.
Ricapitolando, INKEY$ viene solitamente utilizzata per attendere e verificare l'entrata di un carattere dalla tastiera. Ritorna infatti il carattere premuto sulla tastiera al momento della sua chiamata; quando non è premuto (o non è stato premuto) alcun tasto, INKEY$ assegna alla variabile un valore nullo, ed il programma prosegue normalmente. L'esempio che segue potrà chiarirti le idee:

```
10 A$ = INKEY$
20 IF A$ = " " THEN 10
30 PRINT A$
40 END
```

La linea 10 assegna alla variabile A$ il valore del tasto che dovrai premere sul tuo MSX. Se nessun tasto sarà stato battuto, A$ assumerà valore nullo. Sia in un caso che nell'altro il programma non subirà alcuna interruzione o sosta.
La riga 20 controlla quindi il valore di A$: se tale variabile contiene il valore nullo, l'esecuzione riprende alla riga 10, ricominciando il ciclo. L'unica maniera per terminare il programma sarà perciò quella di premere un tasto qualsiasi. Così facendo comparirà inoltre sullo schermo il carattere

battuto (linea 30).
Un tipico utilizzo di INKEY$ è possibile trovarlo in quei programmi che sottopongono l'utente a scelte del tipo:

VUOI CONTINUARE? (S/N)

Simili domande richiedono come risposta la semplice pressione dei tasti S o N: costruendo una struttura a ciclo, simile a quella illustrata in precedenza, è possibile scartare automaticamente tutte le risposte che non rientrano tra quelle ammissibili, evitando magari di riempire lo schermo con inutili ed antiestetici caratteri. Provare per credere.

```
10 A$ = "VUOI CONTINUARE? (S/N)"
20 PRINT A$
30 B$=INKEY$
40 IF B$ = "S" THEN PRINT "SI" : GOTO 70
50 IF B$ = "N" THEN PRINT "NO" : END
60 GOTO 30
70 REM CONTINUA
```

Sintassi della funzione

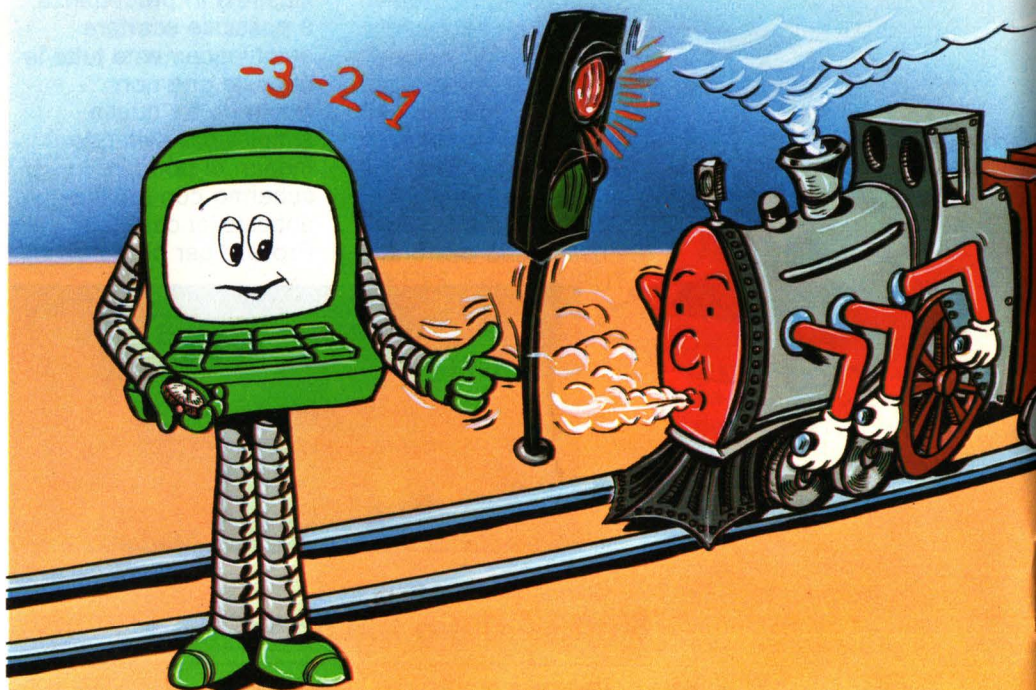
[VARIABILE STRINGA =] **INKEY$**

Come puoi notare, INKEY$ è priva di argomento.

Pausa

Talvolta, nel corso di un programma, può essere utile avere la possibilità di arrestare momentaneamente l'esecuzione delle varie istruzioni.

In alcuni casi possono infatti rendersi necessarie delle pause che permettano all'utente di leggere e valutare una certa visualizzazione o di prendere una determinata decisione.
Una tecnica possibile è quella di imporre al tuo MSX di contare fino ad un certo numero.

Il computer viene in tal modo assorbito da quell'impegno, dando la sensazione di essere in attesa.
Naturalmente quanto più grande sarà il numero a cui deve arrivare, tanto più lunga sarà la pausa.
Se non conosci invece il tempo di attesa, dovrai utilizzare la seguente istruzione:

```
100 A$=INKEY$
110 IF A$ = " " THEN 100
```

Avvisa, però chi utilizzerà il tuo programma di premere un tasto per continuare o interrompere la pausa.

```
90 PRINT "PREMI UN TASTO"
```

Questo comando può essere quindi utilmente adoperato per adeguare il tempo di permanenza sullo schermo delle varie visualizzazioni alle capacità di lettura dell'utente del programma.

KEY ON KEY OFF

Queste istruzioni ti permettono di visualizzare (KEY ON) o di cancellare (KEY OFF) i testi esplicativi dell'effetto dei tasti funzione posti sulla 24ª riga dello schermo.
Come sai, queste scritte appaiono automaticamente all'accensione del computer.
Nel caso però dovessero darti fastidio, nessun problema:

```
KEY OFF
```

seguito da RETURN (ENTER) e la 24ª riga verrà ripulita.
Per ripristinarle:

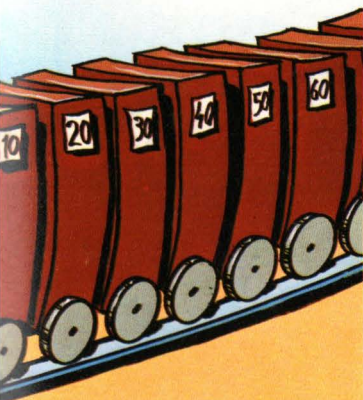
```
KEY ON
```

KEY ON e KEY OFF possono essere utilizzate sia in modo diretto che in modo programma.

```
10 CLS : KEY OFF
20 ...
```

FOR, TO, STEP, NEXT

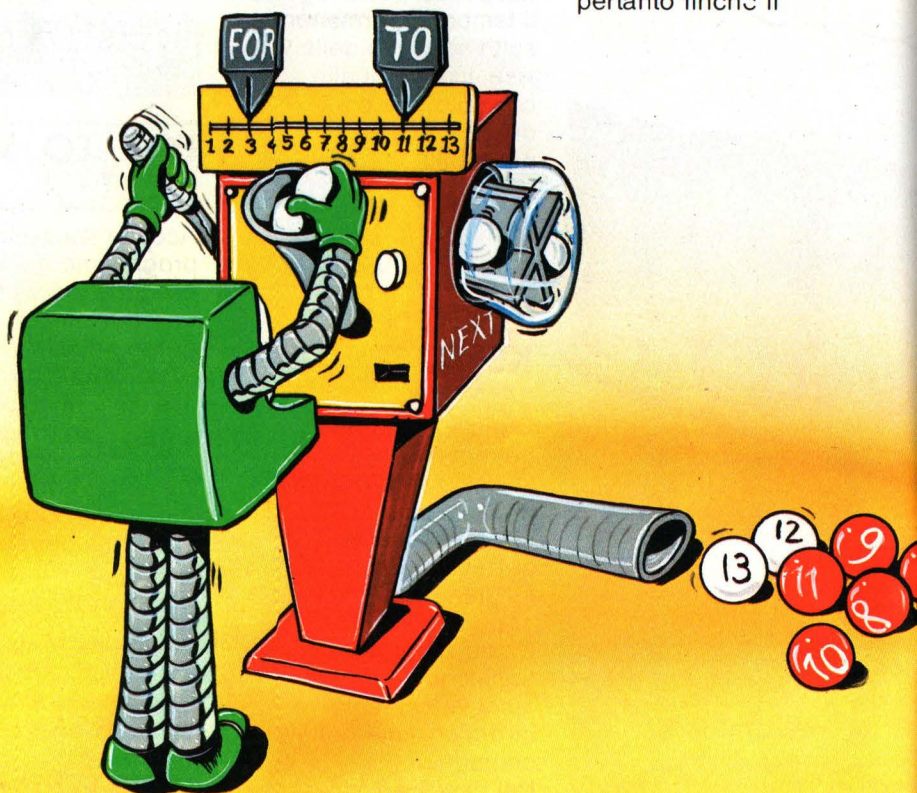
Accade spesso che un programma richieda di eseguire più volte un'istruzione od un gruppo di istruzioni.
Supponi, a titolo di esempio, di voler scrivere un programma che moltiplichi la variabile A per i valori 1, 2, 3, 4, e 5. Una possibile soluzione potrebbe essere:

```
10 LET I = 1: REM I è
   il numero da
   moltiplicare per A
15 LET A = 4
20 PRINT A * I
30 I = I + 1
40 IF I< 6 THEN GOTO 20
```

Si tratta di un tipico esempio di ciclo, cioè una sequenza di istruzioni eseguita un certo numero di volte. Esiste in BASIC una istruzione particolare che ti permette di realizzare i cicli senza ricorrere a strutture complesse. L'istruzione è composta dalle parole FOR...NEXT.
Immagina di dover ripetere una serie di istruzioni per un determinato numero di volte, e precisamente finché un certo contatore C (avente inizialmente il valore S) raggiunga il valore E.
Utilizzando l'istruzione FOR...NEXT potrai scrivere:

```
FOR C = S TO E
```

fai qualcosa

```
NEXT C
```

La prima volta che il ciclo viene eseguito, C è posto uguale al valore di S. Si eseguono quindi tutte le istruzioni.
Arrivati a NEXT il valore di C viene incrementato e confrontato automaticamente con E. Se C risulta minore di E, il ciclo viene nuovamente ripetuto, altrimenti si continua con l'istruzione che segue il NEXT.
La procedura prosegue pertanto finché il

contatore C raggiunge il valore finale di E.
Riprendendo il problema della moltiplicazione si poteva allora dare questa soluzione:

```
10 FOR C = 1 TO 5
20 PRINT A * C
30 NEXT C
```

C si chiama variabile di controllo del ciclo (o contatore) e può assumere un qualsiasi nome (legale) consentito.
Nell'esempio avrai notato che 1 è il suo valore iniziale e 5 è il valore finale o di test.
Nell'istruzione FOR si possono utilizzare anche delle espressioni; per esempio:

```
FOR A= 4+ 5 TO 100/5
```

oppure delle variabili, a patto che siano prima state impostate. Ad esempio:

```
10 LET DA = 5 : LET A = 15
20 FOR C = DA TO A
30 ....
40 NEXT C
```

È inoltre possibile incrementare il valore della variabile contatore con un passo diverso da 1, semplicemente utilizzando la parola STEP (passo) seguita dal valore di cui si vuole incrementare ogni volta il contatore:

```
FOR I= 2 TO10 STEP2
```

I assumerà allora i valori 2, 4, 6, 8, 10.
Il nome della variabile contatore dopo NEXT è facoltativo; se non viene specificato, il tuo MSX lo considera infatti automaticamente collegato all'ultimo ciclo aperto e non ancora completato.
È bene, però, specialmente agli inizi, che tu lo scriva: ciò avvantaggerà notevolmente la leggibilità del tuo programma e la facilità di correzione.
È anche possibile scrivere programmi in cui i cicli contengano al proprio interno altri cicli:

```
┌─10 FOR I = 0 TO 10
│     STEP 3
│ ┌─20 FOR J = 3 TO 9
│ │ 30 ......
│ │ 40 ......
│ │ 50 ......
│ │ 60 ......
│ └─70 NEXT J
└───80 NEXT I
```

Si dice allora che i cicli sono nidificati (ti ricordi di IF...THEN GOTO...).
Naturalmente i contatori dei singoli cicli devono essere diversi. Il ciclo interno ad un altro ciclo deve inoltre essere completamente contenuto nel primo ciclo; non è quindi

possibile sovrapporre parti di cicli. Ad esempio:

```
┌─100 FOR I = 13 TO 20
│┌110 FOR J = 0 TO 10
││ 120 ......
││ 130 ......
││ 140 ......
││ 150 ......
└┼160 NEXT I
 └170 NEXT J
```

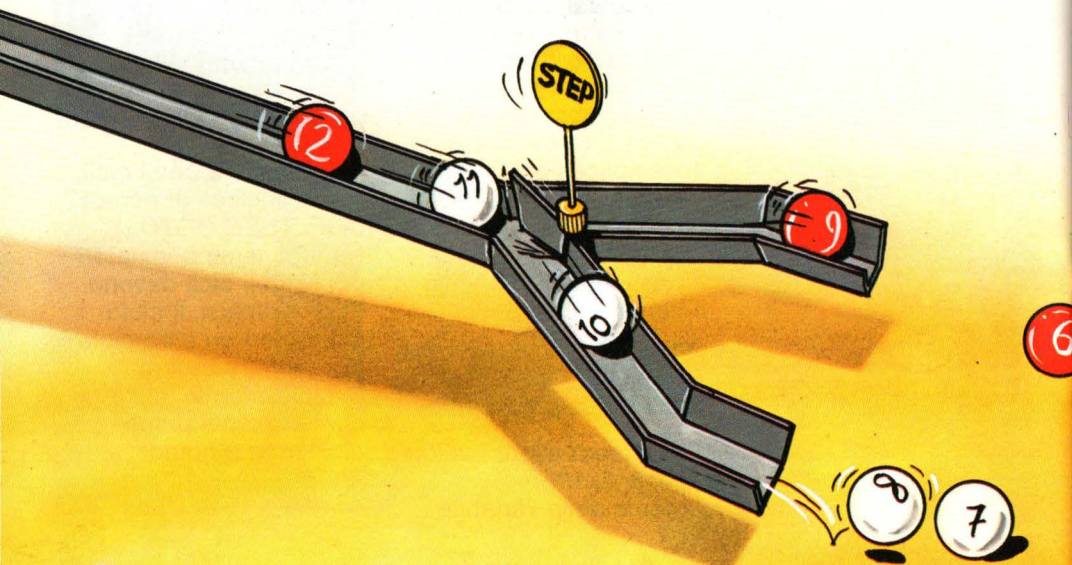
È sbagliato! I due cicli sono infatti sovrapposti.

Molto pericoloso!
Se la variabile indice è inizialmente più grande del valore finale, il ciclo viene eseguito una sola volta. Ad esempio:

```
50 FOR I = 20 TO 5
60 PRINT I
70 NEXT I
```

provocherà una sola volta la visualizzazione sullo schermo del numero 20.
Il passo del ciclo può essere anche negativo:

```
30 FOR X = 13 TO 10 STEP - 2
40 ......
50 ......
60 ......
70 NEXT X
```

In questo caso il ciclo viene eseguito finché X, decrementandosi di due ogni volta, non diventa minore di 10.

Se il valore del passo viene posto uguale a 0, il ciclo si ripete indefinitamente:

```
10 FOR K = 1 TO 10 STEP 0
20 PRINT K
30 NEXT K
```

L'unico risultato di questo programma sarà quindi la continua visualizzazione sullo schermo del valore iniziale di K, cioè 1.

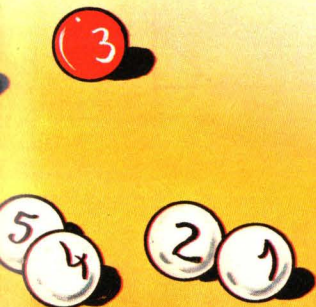
Un errore molto comune è quello di utilizzare un numero di NEXT superiore a quello dei FOR.
In questo caso il tuo MSX risponderà con:

```
NEXT without FOR in...
```

Cerca di evitarlo!

Sintassi del comando

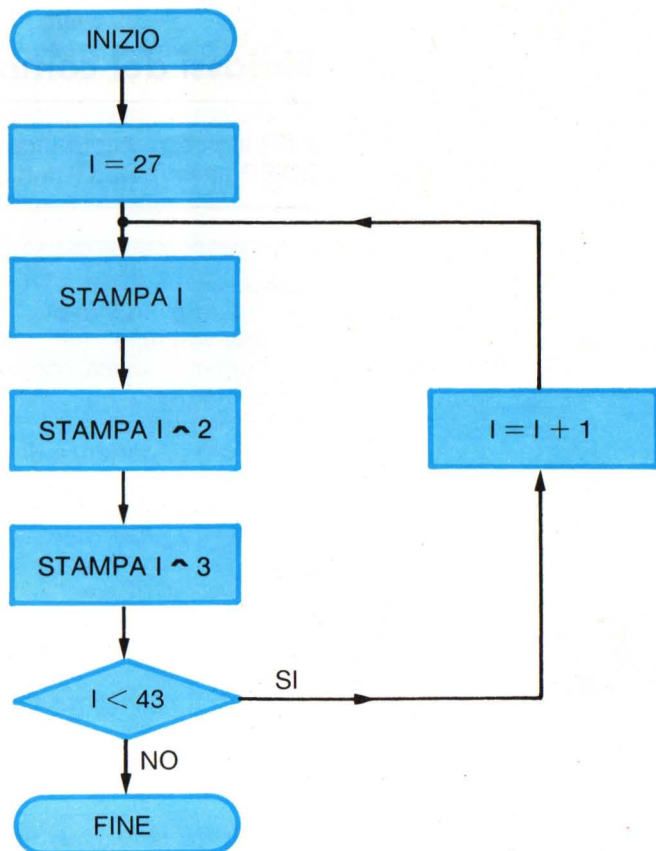
FOR indice = valore iniziale TO valore finale [STEP passo] NEXT [indice]

dove:
- indice è il nome di una variabile numerica usata come contatore
- valore iniziale è il valore di partenza di indice
- valore finale è il valore che l'indice deve uguagliare (o superare) (test)
- passo è l'incremento che ad ogni iterazione subisce indice. Se non è specificato, viene posto pari ad 1 e può essere negativo.

I cicli automatici

Si chiamano cicli automatici tutti quei cicli che eseguono ripetutamente una sequenza di istruzioni facendo uso dell'istruzione FOR...NEXT.
Nella maggior parte dei programmi i cicli automatici sono di impiego talmente utile e frequente da costituire un importantissimo strumento nelle mani di qualsiasi programmatore.
Gli esempi applicativi che seguono ti aiuteranno quindi a chiarire ed approfondire i concetti teorici che già conosci sia riguardo all'uso dei cicli che del comando FOR ... NEXT.

Quadrati e cubi

Come primo esempio consideriamo questo problema: trovare i quadrati ed i cubi dei numeri compresi tra 27 e 43.
Una possibile risoluzione potrebbe essere:

A questo schema a blocchi corrispondono i due seguenti programmi BASIC.
Il primo fa uso di un ciclo controllato (creato cioè "artificialmente" dal programmatore); il secondo di un ciclo automatico:

```
10 LET I = 27
20 PRINT I ^ 2,
30 PRINT I ^ 3
40 IF I < 43 THEN I = I + 1:GOTO20
50 END
```

```
10 FOR I = 27 TO 43
20 PRINT I ^ 2,
30 PRINT I ^ 3
40 NEXT I
50 END
```

Al lettore balza all'occhio immediatamente la migliore leggibilità del programma con il ciclo automatico: in esso si distinguono subito l'inizio e la fine del ciclo, cosa che invece non accade nel primo listato. Anche per il tuo MSX la seconda soluzione è preferibile: l'esecuzione del ciclo è infatti affidata ad una istruzione esplicitamente concepita per risolvere questo tipo di problemi e quindi ad esso più congeniale. Nei casi in cui è possibile (e sono la maggioranza) sarà quindi sempre meglio ricorrere ad un ciclo automatico: il programma - ed il programmatore - ne beneficeranno sotto tutti i punti di vista.

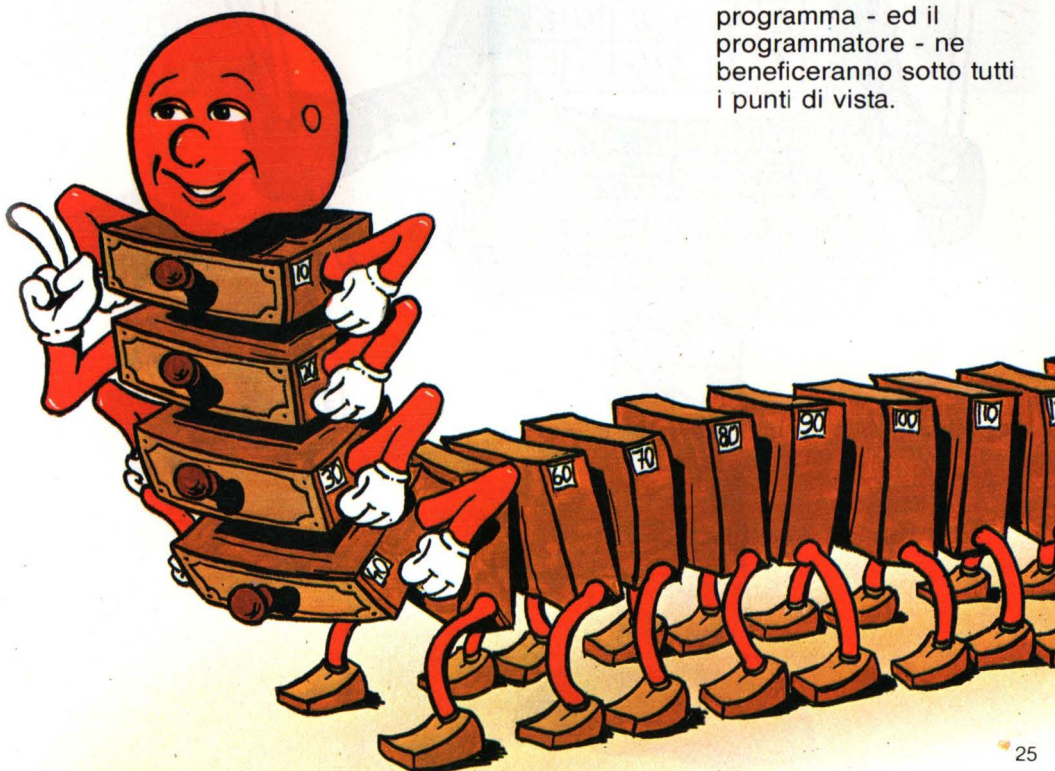
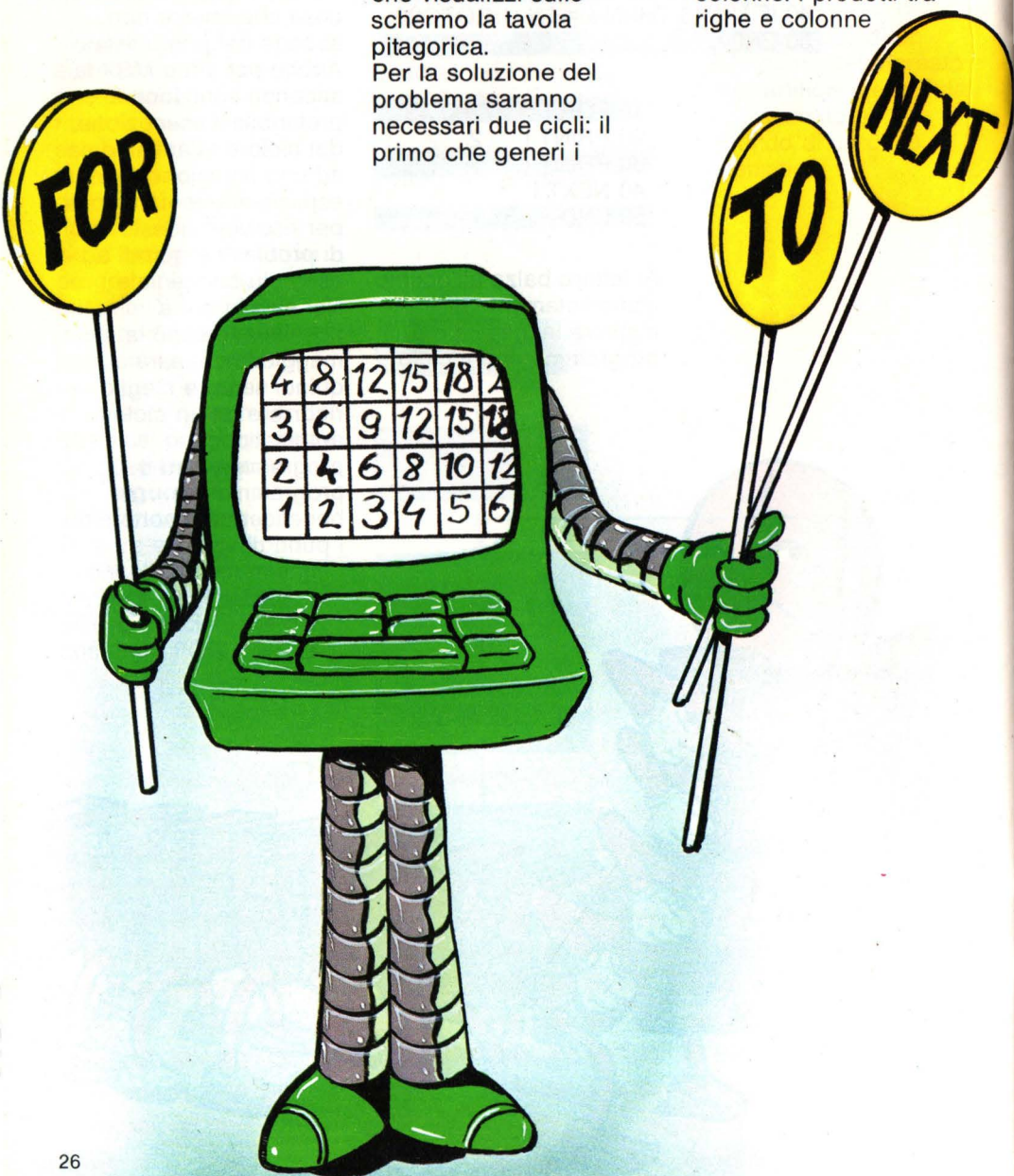
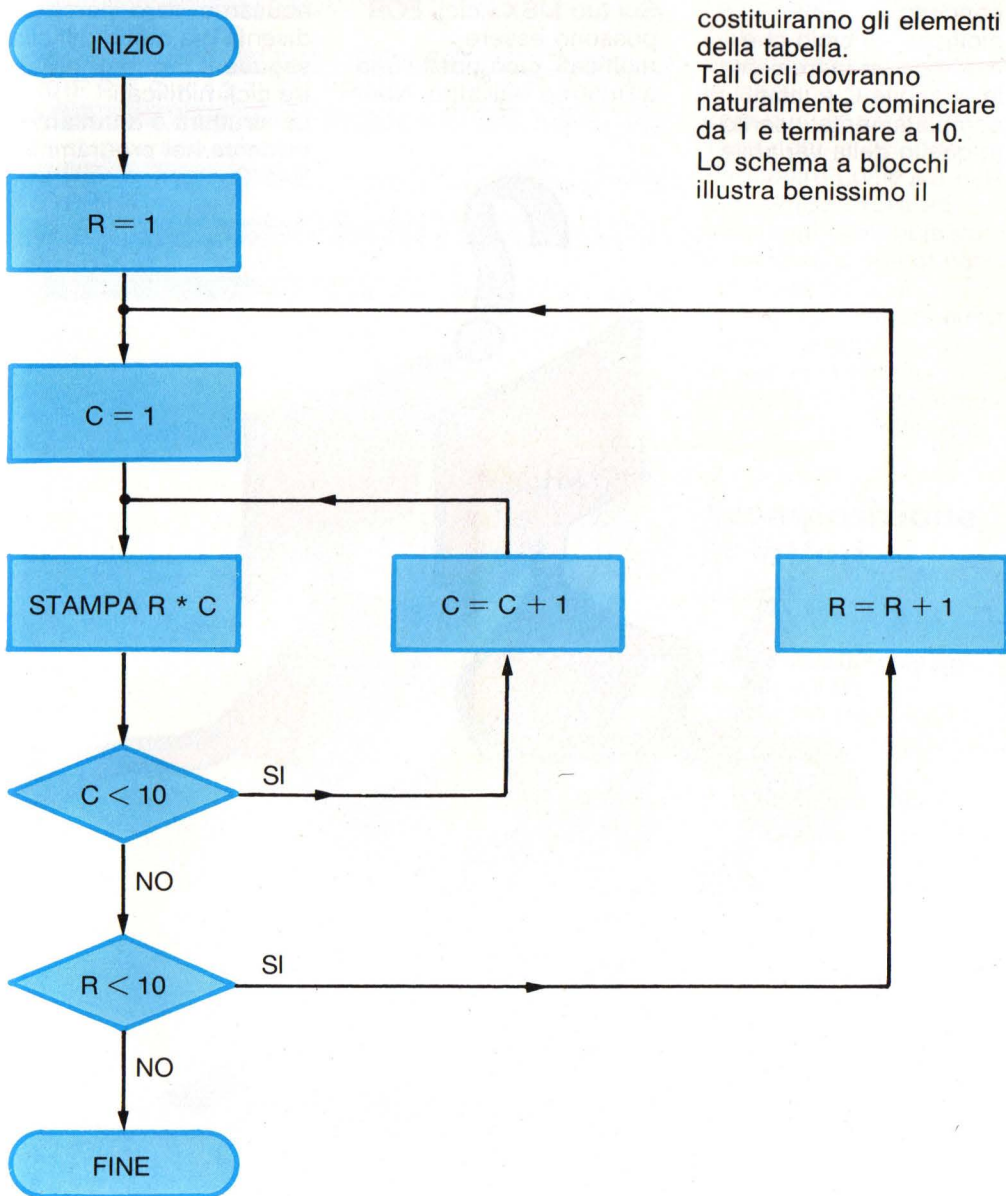
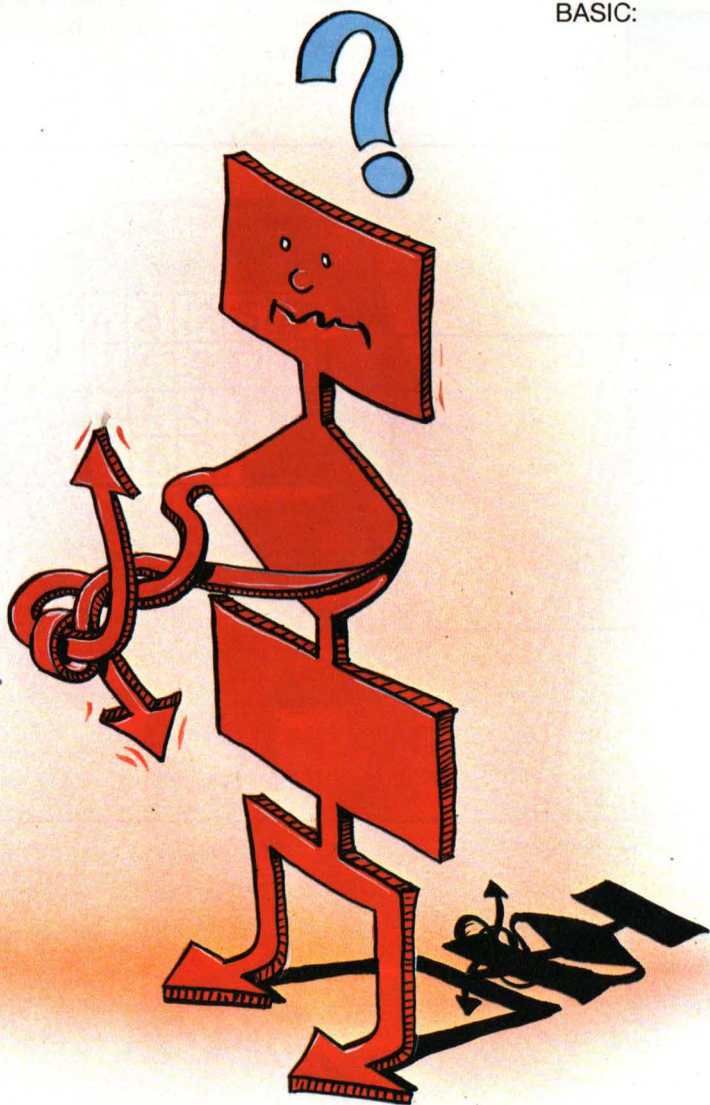



Tavola pitagorica

Come secondo esempio scriviamo un programma che visualizzi sullo schermo la tavola pitagorica.
Per la soluzione del problema saranno necessari due cicli: il primo che generi i numeri corrispondenti alle righe, il secondo alle colonne. I prodotti tra righe e colonne

costituiranno gli elementi della tabella.
Tali cicli dovranno naturalmente cominciare da 1 e terminare a 10.
Lo schema a blocchi illustra benissimo il

concetto di "cicli nidificati"; il ciclo che modifica ed incrementa la variabile C è infatti completamente inserito in quello della variabile R.

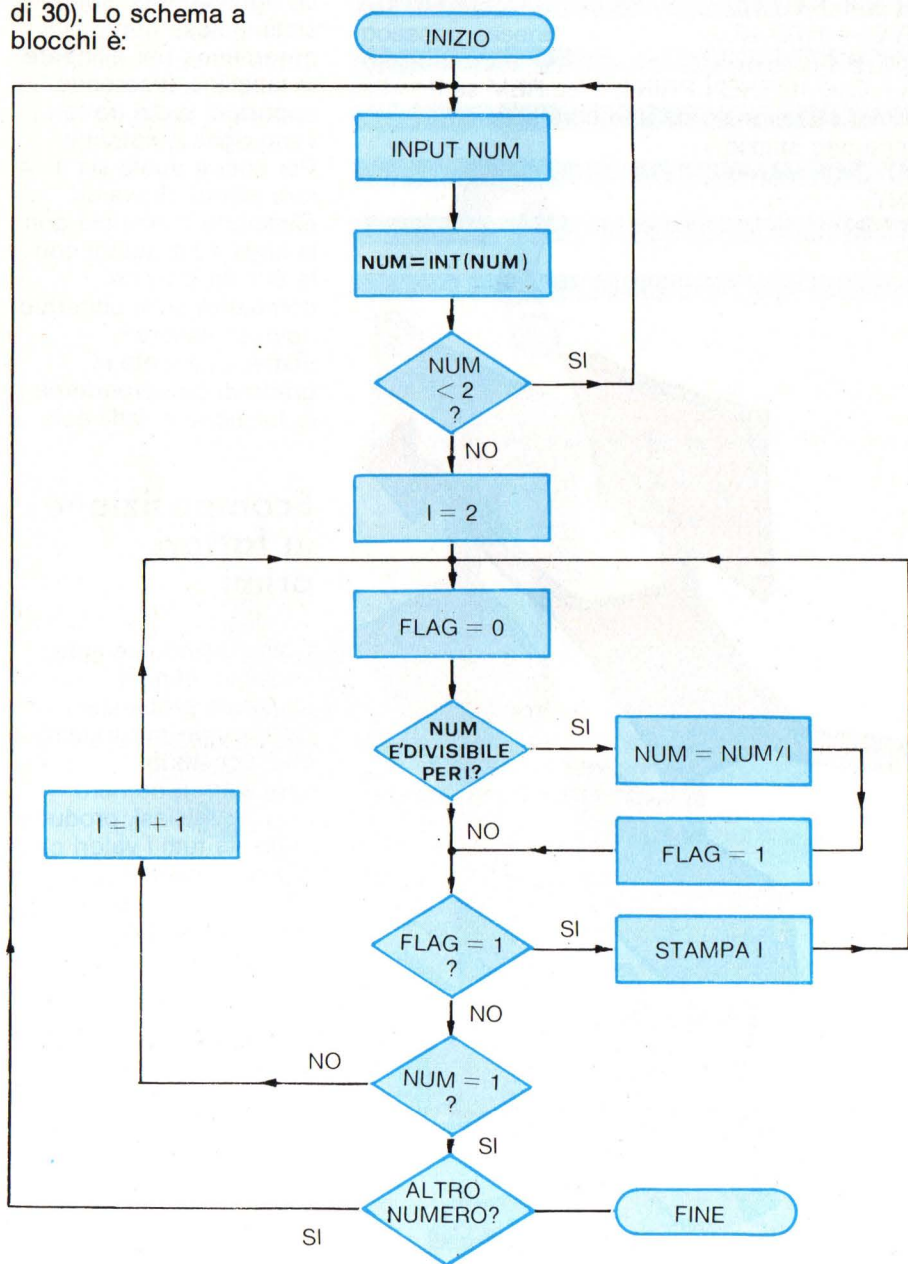
Sul tuo MSX i cicli FOR possono essere nidificati, cioè posti l'uno all'interno dell'altro. Non abusarne, però, perché diventa già molto difficile seguire il flusso di più di tre cicli nidificati.

La struttura è altrettanto evidente nel programma BASIC:

```
10 FOR R = 1 TO 10
20 FOR C = 1 TO 10
30 PRINT R * C;
40 IF R * C < 10 THEN PRINT " "; : REM se il
   prodotto è composto da una cifra sola, allora
   stampa uno spazio
50 NEXT C
60 PRINT
70 NEXT R
80 END
```

Le righe 40 e 60 sono state incluse nel programma per allineare le tabelline, inserendo opportuni spazi tra le varie righe e colonne. Per capire quale sia il loro effetto, prova ad eliminarle (comincia con la linea 40 e quindi con la 60): da ciò che comparirà sullo schermo dovresti essere immediatamente in grado di comprenderne la funzione e l'efficacia.

Scomposizione in fattori primi

Come ultimo esempio vediamo infine il seguente problema. Scrivere un programma che, accettato in ingresso un numero intero qualsiasi, produca in uscita tutti i valori che di tale numero sono divisori primi (ricordati che i numeri primi sono quei numeri divisibili solo per 1 e per se stessi).
Così, battendo il numero 30, il risultato dovrà essere 2, 3, 5 (i divisori

di 30). Lo schema a blocchi è:

Ed il corrispondente listato BASIC:

```
10 CLS : INPUT "numero = " ; NUM
20 NUM = INT (NUM) : IF NUM < 2 THEN RUN
30 CLS : PRINT "i fattori primi di" ; NUM; "sono:"
40 FOR I = 2 TO NUM
50 FLAG = 0
60 IF NUM/I = INT (NUM/I) THEN NUM = NUM/I : FLAG
   = 1 : REM FLAG diventa 1 se NUM è divisibile per I
70 IF FLAG =1 THEN PRINT I : GOTO 40
80 IF NUM = 1 THEN 100 : REM Significa che NUM è stato diviso per se stesso
90 NEXT I
100 PRINT "altro numero ? (s/n)"
110 R$=INKEY$ : IF R$= " " THEN 110
120 IF R$ = "n" THEN END
130 IF R$ = "s" THEN RUN
140 GOTO 110
```

Il programma comincia accettando solo la parte intera del numero e verificando che il valore del numero battuto sulla tastiera non sia inferiore a 2; in caso contrario ne domanda in ingresso uno nuovo.
Dalla linea 40 comincia la vera e propria fase di esecuzione e risoluzione del problema: se NUM (il numero battuto in ingresso) è divisibile per I (linea 60), allora FLAG assume valore 1.
FLAG è una variabile adibita ad indicatore: quando assume valore 1 significa che I è un divisore di NUM e quindi va stampato. Se invece vale 0, I non è divisore di NUM e bisogna incrementarlo di uno.
Man mano che il ciclo prosegue NUM diventa sempre più piccolo; alla fine assumerà valore 1.
A quel punto il problema sarà risolto: sullo schermo saranno infatti comparsi tutti quei numeri che, moltiplicati tra loro, formano il valore iniziale di NUM.

VIDEOESERCIZI

Confronta questi 2 programmi

```
 5 REM PROGRAMMA 1
10 PRINT "INDOVINA COSA SUCCEDE"
20 PRINT "ALLA FINE DI"
30 PRINT "QUESTO PROGRAMMA"
40 CLS
50 REM QUESTO GIA' LO SAI
```

```
 5 REM PROGRAMMA 2
10 PRINT "INDOVINA COSA SUCCEDE"
20 PRINT "ALLA FINE DI"
30 PRINT "QUEST'ALTRO PROGRAMMA"
40 PRINT CHR$ (12)
50 REM CHE SIA LA STESSA COSA?
```

Scrivi il risultato di:
PRINT ASC (*)

Per mezzo della funzione ASC metti in ordine crescente di codice le seguenti stringhe:

"BIT",
"SUPER COMMODORE",
"HOME COMPUTER",
"GATTO"
"VIDEOBASIC"

N°	STRINGA

Prevedi e scrivi l'output di questo programma. Puoi aiutarti con la tabella del codice ASCII.

```
10 PRINT CHR$ (86) + CHR$ (73) + CHR$ (68) + CHR$ (69);
20 PRINT CHR$ (79) + CHR$ (66) + CHR$ (65);
30 PRINT CHR$ (83) + CHR$ (73) + CHR$ (67)
```

A) cronometra la durata del programma e annotala.
B) cerca ed elimina la linea di ritardo (pausa).
C) cronometra nuovamente e annota il tempo.

```
10 CLS
20 FOR D = 10 TO 90 STEP 8
30 PRINT "DATO"; D
40 FOR P = 1 TO 3000: NEXT P
50 NEXT D
```

A	TEMPO	
B	N° LINEA DA ELIMINARE	
C	TEMPO	